

Integrated Self-Localization and Ball Tracking in the Four-Legged Robot Soccer League

Raúl A. Lastra, Paul A. Vallejos, *Member, IEEE*, Javier Ruiz-de-Solar, *Member, IEEE*

Abstract— this article proposed an integrated solution for the self-localization problem of an AIBO robot, as well as for the ball tracking problem, both in the context of the four-legged RoboCup soccer league. The proposed methodology is based on the joint use of Extended Kalman Filtering and Monte Carlo Particle Filtering. Real experiments that validate the proposed methodology are also presented.

Index terms—Four-Legged Robot Soccer, Extended Kalman Filtering Localization, Monte Carlo Localization.

I. INTRODUCTION

Mobile robotics is a lively and expanding research field, whose main goal is the development of autonomous robots that can freely interact in dynamic environments. Some of the main problems to be solved are the ones of perception, navigation, dynamic map building, self-localization, and mapping and tracking of mobile objects [1]. In this article we focus on the resolution of last two problems, self-localization and mapping and tracking of moving objects, for the robot soccer application domain. In a dynamic environment, estimating and predicting the localization of moving objects in the robot's surround (mapping), as well as tracking them, is the basis for an adequate interaction with the world. On the other hand, by estimating the localization of some already know static objects (landmarks), the robot can estimate its own localization (self-localization). Solving simultaneously the task of self-localization as well as the mapping and tracking of moving objects is fundamental for many robots applications as for example museum guiding of children, object transportation in factories and robot soccer. The adequate solution of these problems depends on a number of factors, as the accuracy of the robot odometry, the kind of available sensors and their accuracy, the availability of static know objects (landmarks) and their characteristics, and the predictability of the object's motion. Although there is no universal solution for these problems, most of the successful solutions rely on the use of the Bayesian parameter estimation approach. As already mentioned, in this work we focus on robot soccer, more specifically on the RoboCup four-legged league, where Sony AIBO robots are employed. This application has some special characteristics than are important in the context of the problems being solved: (i) AIBO robots have four, 3DOF legs, (ii) AIBO odometry

information is very noisy, (iii) a low-resolution color camera with 3DOF is the main sensor employed in soccer, (iv) defined landmarks employed for localization are colored goals and beacons (some teams also employ the field lines information), (v) the ball is orange, (vi) communication between robots and with the game controller (a computer referee) is carried put using a wireless data network (802.11b), (vii) no external sensing or processing is allowed, and (viii) the AIBO limited processing power causes constraints in the that can be implemented (more information about this league in [8]). In this context, the methodology being proposed in this article solves the self-localization problem using information of the robot odometry and visual perceptions of the beacons and goals. Camera images together with the information of the robot head and body pose allows extracting distances and angles relative to the landmarks, all having a known position. This sensorial information is fused with the odometry information to get the best position estimation. Both information sources are fused using a mixture of two very known parameter estimation techniques, Extended Kalman Filtering and Particle Filtering. The mapping and tracking of the ball is determined using the perceptual information of the robot as well as information perceived by other robots and communicated using the wireless network. This second information fusion problem is also solved using Extended Kalman Filtering. No mapping or tracking of other robots is performed, because their recognition is still a non-solved problem for us.

II. RELATED WORK

Different approaches have been used to solve the self-localization problem in the RoboCup four-legged league [6][7], being Particle filtering (Monte Carlo Localization - MCL) and Extended Kalman Filtering (EKF), as well as mixture of both, the most popular and successful. There exist also some teams using fuzzy approaches and pure vision for solving this problem. Tables 1 and 2 summarize the methods used in the RoboCup 2003 and 2004 by the 24 teams participating each year in the league (in 2004 there is available information only form 17 teams).

Kalman Filtering is a very well-known technique for parameter estimation, being its main drawback the linearity and Gaussianity assumptions. EKF extends the Kalman Filtering idea by linearizing the measurement and plant (in this case the robot) models, but still has the assumption of Gaussianity [2]. On the other hand, particle filters overcomes the drawbacks of assuming linearity and Gaussianity, by implementing a “factored sampling” of the

Department of Electrical Engineering, University of Chile. E-mail: {rlastra,pavallej,jruizd}@ing.uchile.cl
This research was funded by the research project CONICYT/BMBF 2003-4 -146.

processes' conditional densities [1]. Particle Filtering is very popular in the Computer Vision community where the most employed implementation is called Condensation [1]. In the mobile robotics community the most used implementation is the Monte Carlo Localization – MCL algorithm. The sensor resetting idea was proposed in [4] for improving the MCL results, by introducing in the model, solutions obtained directly from the sensors (re-sampling step). Since then, the sensor resetting idea has been employed by many teams for mixing different localization methods in the MCL algorithm, and also for proposing new MCL variants (Mixture MCL and Adaptive MCL). For instance, in the RoboCup 2003 the German Team employed this idea for implementing its localization module, in which the MCL is mixed with other two localization methods, Single Landmark (SL) self-localization and lines-based self-localization. In our case we also use the MCL for mixing three self-localization methods: MCL itself, SL and EKF. Regarding mapping and tracking of the ball, in the RoboCup 2004 the Huskies team implemented a mixed approach for self-localization, and ball mapping and tracking based on a mixture of EKF and MCL [3]. Some important characteristics of this approach, which distinguish it from ours, are: (i) the robot pose is influenced by ball observations (not in our case), and (ii) the resulting algorithm is very complex requiring about 300-500 particles (70 in the case of the approximate implementation), which not allows an easy implementation in the AIBO processor (in our case just 20 particles are required!).

Table 1. Self-Location methods implemented in 2003 by the 24 classified teams of the RoboCup four legged league.

ROBOCUP 2003		
METHOD	N° TEAMS	Percentage
MCL	11	45,8 %
EKF	4	16,66%
MCL and EKF	0	0%
FUZZI LOGIC	2	8,33%
VISION	5	21,73%
Subjective Map generation	1	4,16%
Stochastic Gradient Descent	1	4,16%

Table 2. Self-Location methods implemented in 2003 by 17 of the 24 classified teams of the RoboCup four -legged

ROBOCUP 2004		
METHOD	N° TEAMS	Percentage
MCL	10	58,88%
EKF	2	11,76%
MCL and EKF	2	11,76%
FUZZI LOGIC	1	5,88%
VISION	2	11,67%

III. PROPOSED METHODOLOGY

A. Overview

Our robot soccer software library (U-Chile 1 library) is divided into five task-oriented modules: *Vision*, *Localization*, *Low-level Strategy*, *High-level Strategy* and *Motion Control*. The vision, motion control and low-level

strategy modules operate in each robot locally. The localization module is distributed, it operates on each robot and a global estimate of the overall localization is generated in a distributed fashion. The high-level strategy module is also distributed, and allows the sharing of global information among the robots. In figure 1 is shown a block diagram of our *Localization* module and how it interacts with the other modules of our software library. *Vision*, which is in charge of the robot perceptions, obtains the information of the perceived objects of interest, in this case the landmarks and ball positions (distance and angles). *Motion Control*, which is in charge of the low-level robot control, gives the odometry information obtained directly from the instruction list (orders) sent to the motors. By joining these both information sources, vision and odometry, together with the ball estimation information communicated by other robots, the *Localization* module estimates the robot pose and the ball pose. These estimations are communicated to the local *Low-Level* and *High-Level Strategy* modules, and to the *Localization* and *Strategy* modules of the teammates. For the estimation of the robot pose we combine the representation capabilities of the Monte Carlo Filters with the efficiency and accuracy of Kalman Filters, *MC* and *EKF Self-Localization* blocks, respectively. Using EKF we obtain a fast convergence time, and using MC we obtain stability and robustness. There are some game situations in which the robot moves itself to a given position where it can see only one landmark. In this case neither MC nor EKF can estimate adequately the robot pose. However, using a special algorithm that uses the information obtained from a single landmark and some geometrical assumptions [9], is still possible to obtain a reasonable pose estimation. This algorithm is implemented in our *SL* (Single Landmark) *Self-Localization* block. Finally, all robot pose estimation hypothesis are mixed in the re-sampling step of the MC algorithm (altogether only 20 particles are employed). The ball pose estimation is implemented in the *EKF Ball Localization* block, which fuses ball perceptions with ball estimation information communicated by other robots. In our implementation the estimated robot pose influences the ball pose estimation, because both are implemented using a single EKF whose state vector includes robot and ball information. However, we impose the constraint that the ball estimation cannot influence the pose estimation (in the Jacobian matrix of the observational model some values are set to 0).

B. Self-Localization

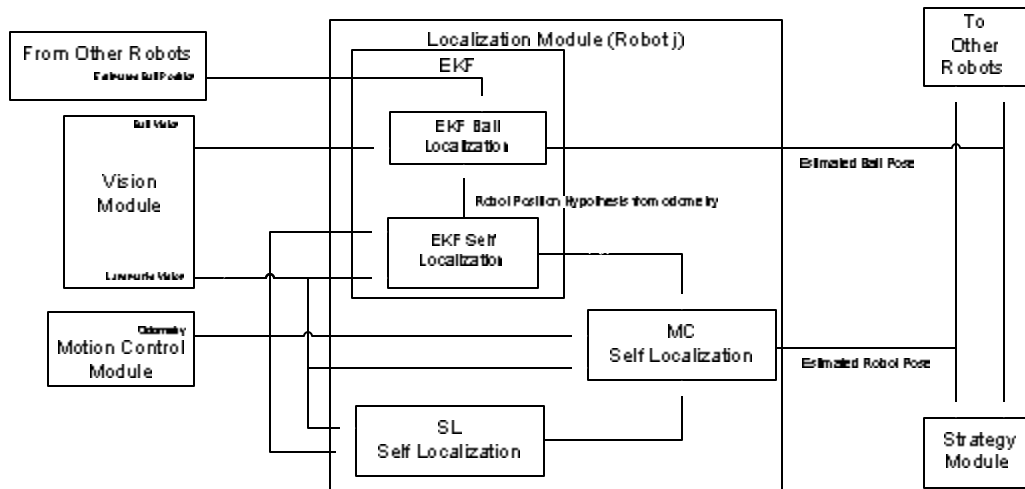
1) Extended Kalman Filter

The state vector $x \in \mathfrak{R}^3$ corresponds to the robot pose and is defined as

$$x = [x_{robot} \quad y_{robot} \quad \mathbf{q}_{robot}]^T \quad (1)$$

The robot process is modeled with a non-linear function f , which depends on the past state vector x_{k-1} and the control orders sent to the robot motors u_{k-1} :

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \quad (2)$$



correctly and the error variance is large (18°). Taking into account these considerations the error table was filled using the measured distances shown in table 3 (angle errors are either 2° or 18°).

Table 3: Object distance dispersion (in centimeters).

Real Distance	50	100	150	200	250	300	350	400	450
Landmark Distance Dispersion	1	1	3	6	20	30	35	40	50
Goal Distance Dispersion	1	2	3	4	8	10	13	17	25
Ball Distance Dispersion	1	3	7	13	19	30	38	50	61

For each estimated robot pose we can compute $h(x_k, 0)$ according to our geometric model. Then, we must calculate $H_k \in \mathfrak{R}^{m \times m}$, that is the Jacobian matrix of partial derivatives of h with respect to x :

$$H_{[i,j],k} = \frac{\partial h_i}{\partial x_{j,k}^-} = \frac{h_i(x_{j,k}^- + \Delta, 0) - h_i(x_{j,k}^- - \Delta, 0)}{2\Delta} \quad (7)$$

with $z_{i,k} = h_i(x_k^-, 0)$.

The Kalman gain is then computed as:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \quad (8)$$

with $V_k \in \mathfrak{R}^{3 \times 3}$ the Jacobian matrix of partial derivatives of h with respect to v , set to the identity matrix in our case. Finally, robot pose estimation and the covariance matrix are computed:

$$x_k = x_k^- + K_k (z_k - h(x_k^-, 0)); P_k = (I - K_k H_k) P_k^- \quad (9)$$

2) Single-Landmark method

This method was implemented based on [9]. The rationale behind the method is: when we have a one landmark distance this measure permit describe a circle of positions around the landmark and if we consider the last position estimated is possible choose the nearest position in the circle like a “virtual position” then we can interpolate these position to obtain a position estimated in this iteration. The robot orientation will be obtained in the same way, interpolating the last orientation with the orientation obtained of the angles measurements to the landmark image. This is illustrated in figure 2.

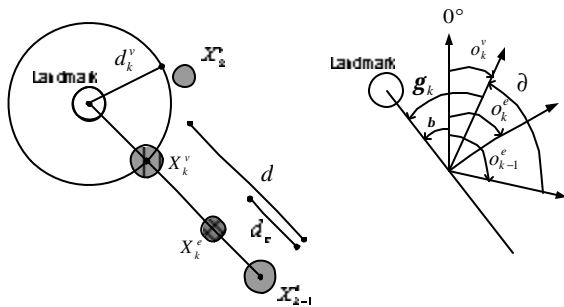


Fig 2. Single landmark localization distance update (left) and angle update (right) X_k^r : Real robot position in time step k. X_k^v :

Virtual robot position in time step k. X_k^e : Robot position estimated in time step k. d_k^v : Measured landmark distance in time step k. d_c : d multiplied for the distance correction factor. g_k : Measured angle to landmark in time step k. b : Angle between straight line from landmark to current position and x axis. o_k^v : Virtual robot orientation in time step k. o_k^e : Robot orientation estimated in time step k. d : Difference between last robot orientation estimated and virtual robot orientation in time step k. The distance correction factor used was 0.5. The same factor was used to estimate the robot orientation.

3) Monte Carlo Particle Filter

We use the classic MCL method with a population of 20 individuals. The individuals' pose are actualized using the odometry information in the same way as in the EKF. MCL incorporate a probabilistic component to each movement model:

$$\begin{bmatrix} x_k \\ y_k \\ o_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ o_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta o \end{bmatrix} + \begin{bmatrix} fx * (1 - p_j) * a \\ fy * (1 - p_j) * b \\ fo * (1 - p_j) * c \end{bmatrix} \quad (10)$$

The value p_j is called the weight of each individual and is a measurement of how probably it is. The values a , b and c are uniformly distributed random variable with range $[-1, 1]$, fx , fy and fo are constant mutation factors. The following step is to calculate weights for each individual using the sensorial information. The weight of individual j is calculated from the following equation:

$$p_j = \prod_{i=0}^{m-1} e^{-\frac{(h(x)_i^d - d_i)}{\sigma_d}} * \prod_{i=0}^{m-1} e^{-\frac{(h(x)_i^o - o_i)}{\sigma_o}} \quad (11)$$

with $h(x)_i^d / h(x)_i^o$ the measure distance/angle to the landmark i . We limited the weights change as:

$$p_j = \begin{cases} p_j & \text{if } (p_j - p_{j-1}) < 0.1 \\ p_{j-1} + 0.1 & \text{if } (p_j - p_{j-1}) > 0.1 \\ p_{j-1} - 0.05 & \text{if } (p_j - p_{j-1}) < -0.05 \end{cases} \quad (12)$$

In this way we prevent large changes in the weights, produced for instance by vision noise.

To improve MCL performance we use the sensor resetting idea, e.g. SL and EKF method gives solutions that are introduced in MCL. Using the EKF and SL solutions, more different individuals are generated. In our approach we use 5 individual generated from EKF (1 individual is equal to the EKF solution), and 1 individual equal to the SL solution. The weight is calculated for each individual, and then they continue the algorithm in the same way of any other individuals. All the weights are normalized, because the meaning of these weights is nearly related with the probability density function and will be essentials to the next step called re-sampling. The re-sampling step is a probabilistic way to choose some individuals to the population, e.g. the better individual have much

probabilities to keep in the next population. The re-sampling algorithm we implemented is described in figure 3.

```

while (j < PopulationSize) {
  while (i < Population Size) {
    S = Random_Value;
    if (Weight(i) < S)
      Individual [ i ].KeepNextPopulation [ j ];
    j = j + 1;
    break ;
  }
  i = i + 1;
}
i = 0;
}

```

Figure 3: Re-sampling algorithm.

Finally we perform a probability density function for the system state represented for the individuals of the population, but from this function we must obtain an only one solution. We do that calculating the average of all individual considering their weights:

$$\bar{x} = \sum_{i=0}^{P_Size-1} x_i * p_i; \bar{y} = \sum_{i=0}^{P_Size-1} y_i * p_i; \bar{\sigma} = \tan \left(\frac{\sum_{i=0}^{P_Size-1} \sin(\theta_i)}{\sum_{i=0}^{P_Size-1} \cos(\theta_i)} \right) \quad (13)$$

And the belief in the pose estimation is calculated as the dispersion of the individuals.

$$s = \sqrt{\sum_{i=0}^{P_Size} ((\bar{x} - x_i)^2 + (\bar{y} - y_i)^2) * p_i} \quad (14)$$

C. Ball-Localization

The implemented EKF allows us include more state variables to the state vector. Considering that, we included the ball position in the state vector, and compute the ball vision like any other landmark. The sensorial information gives us a ball position relative to the robot but, if we know the robot position is possible calculate the ball position in the field. In this way the new state vector $x \in \mathcal{R}^5$ includes the robot pose and the ball position. The measurements relative to the robot are transformed in a field position and are computed in the z vector. The function $h(x)$ includes directly the ball position and all the operations realized in the EKF for self-localization are computed in the same way here. However, in this case is necessary to manipulate the process to get the robot pose non-dependent of the ball position. For obtaining that, when the H matrix is built we must to set in zero the values of:

$$\frac{\partial h_d^b}{\partial x_{robot}} = \frac{\partial h_d^b}{\partial y_{robot}} = \frac{\partial h_d^b}{\partial \mathbf{q}_{robot}} = \frac{\partial h_q^b}{\partial x_{robot}} = \frac{\partial h_q^b}{\partial y_{robot}} = \frac{\partial h_q^b}{\partial \mathbf{q}_{robot}} = 0$$

Other important factor is the communication. In many game situations the ball is not seen by a robot. Considering that, we improve the ball localization using the wireless communication ability of the AIBO robots. The communication information is computed in the vector z as any other sensor measurement. If one robot doesn't see the ball, it transmits the information in any case but sends an

indicator too and its information about the ball is ignored in the receptor robot and not including in z vector.

$$z_k = [d_1 \ o_1 \dots \ d_{m/2} \ o_{m/2} \ x_b^j \ y_b^j \ x_b^{j \neq j} \ y_b^{j \neq j} \dots \ x_b^{i \neq j} \ y_b^{i \neq j}]^T \quad (15)$$

IV. EXPERIMENTS AND RESULTS

We performed basically two kinds of experiments, based in two kinds of situations, the kidnapped problem and the location following a path.

The ability of follow a path and keep itself localized is the most important problem, because that situation is constantly happening during the game. To realize measurement of the method performance for these cases we defined five points in the field, in the center and in the corners, the robot walk towards these points, when the robot stand in the point wanted it alternate to other point. In this process a timer implemented in the code stop the robot in random time steps. When this happen internally we caught the position estimated. And we can measure the real position. After this the robot continues its walk. This way, we can take estimation error in a random time step. Beside, while the robot is following its path the ball is in a fixed position and this way we can calculate the ball position estimation error. Was computed 16 points and the average of errors was calculated. The results are presented in table 5. The kidnapped convergence time was measurement starting in several positions where the localization method was converged. Then we paralyze the localization method until we put the robot in other different position. After this we activate the localization method and a timer to measure the time of convergence. We use a boundary of convergence in a circle of twenty centimeters ratio. Beside to calculate the distance to the real point we compute a measure of orientation differences and that was adding to the Euclidian distance. $\Delta = \sqrt{(x - \bar{x})^2 + (y - \bar{y})^2} + 10|o - \bar{o}| < 20$. The kidnapped convergence time was measurement just moving the ball to several positions in front of the robot. Considering that the error in the estimation of ball position is greater than the self position we use thirty centimeters for measure the time convergence for ball position. When the position estimated is inside of this circle the timer stop and registers the time.

Table 4: Localization average error, following a path.

	$\Delta \bar{x}$ [cm]	$\Delta \bar{y}$ [cm]	$\Delta \bar{o}$ [°]	$\sqrt{(\Delta \bar{x})^2 + (\Delta \bar{y})^2}$ [cm]
Self Location	19,09	20,67	12,67	30,69
Ball Location	24,49	26,49	-	39,19

For the kidnapped experiments were realized 16 experiments in different points. We took the convergence times and calculate an average time. The result at self-localization convergence time was 3,35 seconds and for ball-localization was 2,99 seconds. In the following images sequence is showed the evolution of MCL population, considering individuals inserted from EKF and SL, resolving a kidnapped situation. The robot was collocated in one side of the field until location module was converged. Then the robot was moved to other side of the field and rotated in 180°. Then, the ball was moved in front

of the robot again. In the figure 4.(a) is showed self and ball localization converged in a side of the field. The landmark in the corner is seen to lees distances that the real and SL individual is average between blue individual and the measurement. In the figure 4.(b) the robot was kidnapped and some individuals (SL and EKF) moved to the center of the field, the population is nearly to the initial point but the

selected moved to the center because the individuals in the center have better weights. In figure 4.(c), population is almost complete in the other side of the field, and in figure 4.(d) the method was converged. The figures 4.(e) and 4.(f) shows the convergence of the ball position when it was moved in front of the robot again.

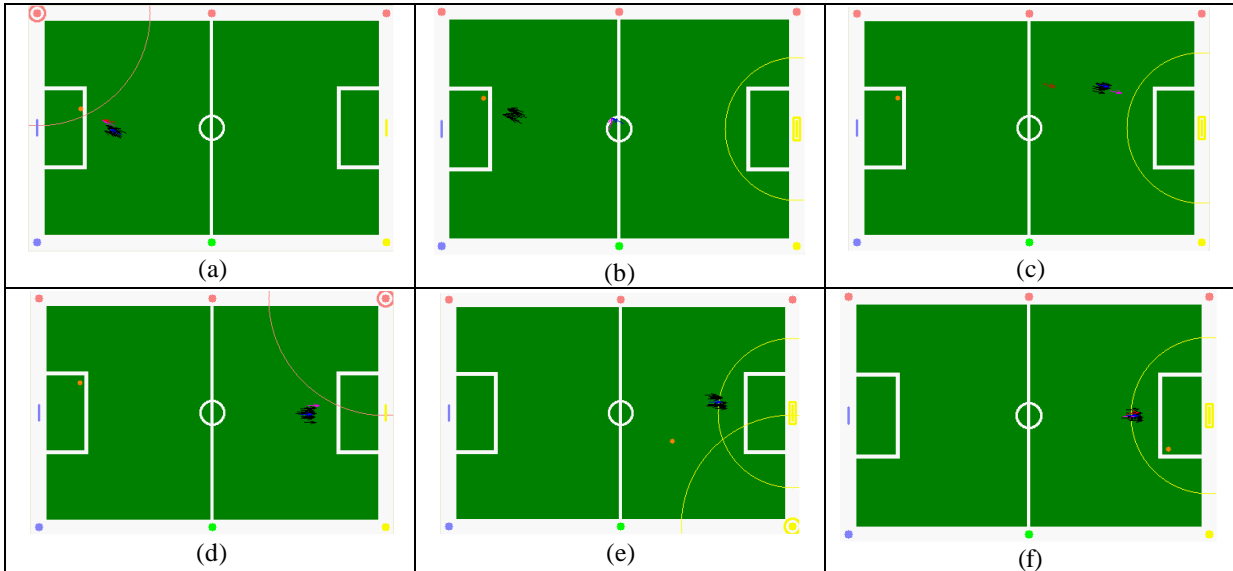


Figure 4: Localization method in a kidnapped situation. The MCL population is colored in black EKF is red and SL is pink, the final result of location module is colored in blue. The color circles on the filed represents the distance measurement to landmark for vision module, while the respective landmark is rounded by a circle.

V. CONCLUSIONS AND PROJECTIONS

In Robocup 2003, none method for ball localization or communication between robots were used, nevertheless, it was used a self-localization module based on MCL method. Nowadays, the localization work was improved with the combination of MCL with EKF and Single Landmark. Using or not communication abilities, the ball localization method has successful results. The combination of the methods is an excellent way to achieve speed, accuracy and robustness in the localization estimation problem. This, because MCL doesn't need the Gaussianity assumptions, it can represent any probabilities density function, meanwhile, the EKF and Single Landmark incorporation, gives rapidity and reactive characteristics. When communication property is incorporated, the ball localization is much better and are generated many possibilities to create a global strategy that improve the performance of the team. Due to the good obtained results using EKF, for futures works, another approach for location problem based on Kalman filter will be considered, "Ensemble Kalman". At the moment, the vision module, detects goals and beacons, but in the game the robot is searching the ball or following the ball, most of the time, in those cases the field lines are seen in more times and of better way. The lines detections in the vision module would improve the general performance of the localization module, incrementing the sensorial information extracted to the environment. The localization module, use several parameters that need to be tuned. At the moment, the parameters are adjusted manually. Is possible, to create

an adaptive method to adjust the localization method parameters. On the other hand, to develop methodologies and systematic ways to measure errors of vision and odometry, will allow the incorporation of factors of correction.

VI. REFERENCES

- [1] M. Isard and A. Blake, "Condensation – Conditional Density Propagation for Visual Tracking", *Int. J. of Computer Vision*, 29, 1, 5–28, (1998).
- [2] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*, Cambridge University Press, 2000.
- [3] C. Kwok and D. Fox, "Map-based Multiple Object Tracking of a Moving Object", *Lecture Notes in Computer Science (RoboCup 2004 Symposium)*, 2004. (in press).
- [4] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modeled mobile robots", *Proc. of the IEEE Int. Conf. on Robotics and Automation – ICRA 2000*, San Francisco.
- [5] J. Ruiz-del-Solar, P. Vallejos, J. Zagal, R. Lastra, G. Castro, C. Gortaris, I. Sarmiento, "UCHile1 2004 Team Description Paper", *Proc. of the RoboCup 2004 Symposium*, July 4 – 5, Lisbon, Portugal (CD Proceedings).
- [6] D. Polani, B. Browning, A. Bonarini, K. Yoshida (Eds.), *Proc. of the RoboCup 2003 Symposium*, July 9 - 11, Padova, Italy. (CD Proceedings).
- [7] D. Nardi, M. Riedmiller, C. Sammut, J. Santos Victor (Eds.), *Proc. of the RoboCup 2004 Symposium*, July 4 – 5, Lisbon, Portugal (CD Proceedings).
- [8] RoboCup Four-Legged League Official Web Site: <http://www.openr.org/robocup/index.html>
- [9] German Team 2003 Technical Report <http://www.robocup.de/germanteam/GT2003.pdf>